the hyperwall

Timothy A. Sandstrom *, Chris Henze †, Creon Levit ‡
Exploratory Computing Environments Group,
NASA Ames Research Center

Abstract

This paper presents the hyperwall, a visualization cluster that uses coordinated visualizations for interactive exploration of multidimensional data and simulations. The system strongly leverages the human eye-brain system with a generous 7x7 array of flat panel LCD screens powered by a beowulf cluster. With each screen backed by a workstation class PC, graphic and compute intensive applications can be applied to a broad range of data. Navigational tools are presented that allow for investigation of high dimensional spaces.

Keywords— **Scientific visualization, multivariate analysis, linked views, brushing**

1 Introduction

That data are growing in size and complexity is self evident, no more so than when it comes to multidimensional/multivariate (MDMV) data. Scientific and Information visualization literatures are filled with research delving into the issues associated with this data explosion. A common theme emerges: sooner or later, machines run out of some precious resource, be it CPU, graphics, screen real estate, memory, or disk bandwidth. Screen space, for instance, has been a limiting factor in MDMV visualization systems ever since the first brushed scatterplot matrix [1, 2]. One can only display so many windows, can only present so many variables in a single view before reaching a point of diminishing returns.

We seek to interactively explore large, MDMV datasets and families of parameterized simulations. Towards this end we have assembled a system using a combination of commodity hardware and custom software known as the 'hyperwall'. Combining a phalanx of pixels and processors, we seek to overcome some of the graphics and computational limitations found in many MDMV visualization systems and work towards a true problem solving environment where many tools can be brought to bear on a given problem at once.

There are many approaches to MDMV visualization, the goal typically being to visually summarize and interact with the data searching for trends and relationships. Tools such as XGobi [3] and XmdvTool [5], use techniques such as scatterplots, glyphs, and parallel coordinates to display MDMV data in lower dimensional projections. Direct manipulation techniques such as interactive brushing are used to find relationships between variables. These packages draw on the works of authors such as Tukey and Cleveland, providing a rich set of the classic statistician's tools to bring to bear on the data. Other research has focused on multiple, coordinated views or visualizations of related data. North and Shneiderman have revealed the efficacy of these techniques across a broad spectrum of problems. Finally, a number of tools have taken a spreadsheet approach to MDMV visualization, where the layout of the views have inherent meaning, implying location and allowing navigation in a given high dimensional space. This approach also allows a high degree of coordination between views, for instance when a given modification or operation is applied to all visuals in a column or some other subset of the matrix.

The hyperwall exists at the confluence of these streams, combining spreadsheet metaphores, direct manipulation, and multiple linked coordinated views.

2 System Architecture

The core of our system is a 49-node beowulf cluster. Each node uses an nVidia GeForce4 graphics card to drive one of the LCD flat screen monitors arranged in a 7x7 matrix on a custom rack. This gives us around 64 million pixels spread across some 55 square feet of screen real estate. All nodes run without a keyboard or mouse, and custom software has been written to allow a user to interact with the nodes. This software, hyperx, captures all mouse

*AMTI Inc., sandstro@nas.nasa.gov
†AMTI Inc., chenze@nas.nasa.gov
‡NASA, creon@nas.nasa.gov

and keyboard events on a master node and broadcasts the identical X event stream to any or all nodes allowing user control of any GUI-based software running on the cluster.

3 Spreadsheet-Based Visualization

Since we lay our visualizations out in a matrix, our system is related to spreadsheet-based visualization systems [6, 7, 8] and gains many of the inherent benefits thereof. The position of a visualization in our matrix of screens can be directly related to a location in a high dimensional space, providing the user with a necessary context for understanding and navigating MDMV spaces. For instance, Fig 2 shows a parameter study of the Reusable Launch Vehicle. Each row displays a different angle of attack, each columm a different Mach number. The user knows, at a glance, the parameters associated with the dataset in each view.

Like other spreadsheet-based systems, the simultaneous display of related visualizations facilitates a broad range of primarily visual activities such as comparing and contrasting related images, tracking features between timesteps, and finding patterns amidst the complexity of a family of bivariate scatterplots. Furthermore, with aggregate visualizations (using possibly several different applications), we can explore visualization space as well as data space, looking at our data in a number of ways at once. Our system provides well for these visual tasks by providing high resolution views of all visualizations and allowing for a high degree of interactivity with these views by the user.

3.1 Caveats

One way in which we differ from spreadsheet-based visualization systems is that, in general, we do not have a built in programming language. This would allow one, for instance, to calculate the difference between the scalar fields on two nodes, and display the results on another. However, this might require a high degree on integration between software on the master node and software running on the nodes. While we have written applications that are as tightly coupled as this, node application software can often be run unmodified. This loose coupling greatly extends the number of applications we can use on the cluster especially those for which we do not have source. Alas, some features may require code modification no matter what. For example, synchronized animation across multiple nodes will typically involve some external synchronization mechanism probably not anticipated in a given application.

4 Coordinated Visualizations

In our system, nodes either work together in order to display one scene (like a powerwall), or they each individually display a separate (possibly related) scene. Coordination in the powerwall sense requires that all nodes render the same scene at the same time with a shared set of viewing transformations. At the other end of the spectrum, each node displays possibly a different dataset, or the same dataset using a different rendering parameter, or even a completly different visualization technique. Coordination in this case may mean that all the nodes have the same viewing transformations, or that the same colormap is used across the different datasets. In any case, we achieve coordination of views by providing each node with the exact same stream of X events.

4.1 Using X to interact with the nodes

Using the X Test extension distributed with X11R6, we can send simulated mouse and keyboard events to X servers running on any of the nodes. Using custom software, called hyperx, a user can sit at the master node and interact simultaneously with any number of nodes in the matrix. This allows one to change such things as view perspective, color mapping, visualization type, or any parameter of a visualization accessible via a GUI. Imagine interactively changing a cutting plane position through 49 different datasets at once, and you begin to see the possibilties of such as system. Another nice feature is the ability to move the mouse and keyboard around screen to screen as if you are interacting with one very large virtual desktop.

4.2 Issue: Maintaining View Coherence

In powerwall mode, when a group of nodes are cooperating to show a single scene, all the nodes must agree upon the current set of transformations. Similarly, when groups of nodes independantly render related objects, we typically want to have the same viewpoint across all of them. Transformations or viewpoints are usually modified via the mouse or keyboard. Thus, we can often achieve view coherence by sending the exact same event stream to each node though there are again some subtleties that may require modifications to software running on the nodes.

When adjusting the view with the mouse, an X-based application will typically pass through the main loop many times as the mouse is dragged. Since we want all ganged nodes to respond identically, any source of asynchronicity in the event production or consumption must be hunted down and removed. Thus for example, all event compression must be turned off. Other sources of asynchronicity are X workprocs, and callbacks from I/O events on sockets registed via XtAppAddInput. Self generated X events can be another source of disparity in the event stream and hence lead to divergance of transformations and viewoints.

Once these and other sources of randomness in either the event stream production or consumption have been removed, the remaining issue is that of graphics throughput. Suppose, as you are drawing your scene across four nodes ganged in a 2x2 array, one node's portion of the view frustum happens to contain a bulk of the polygons in the isosur-

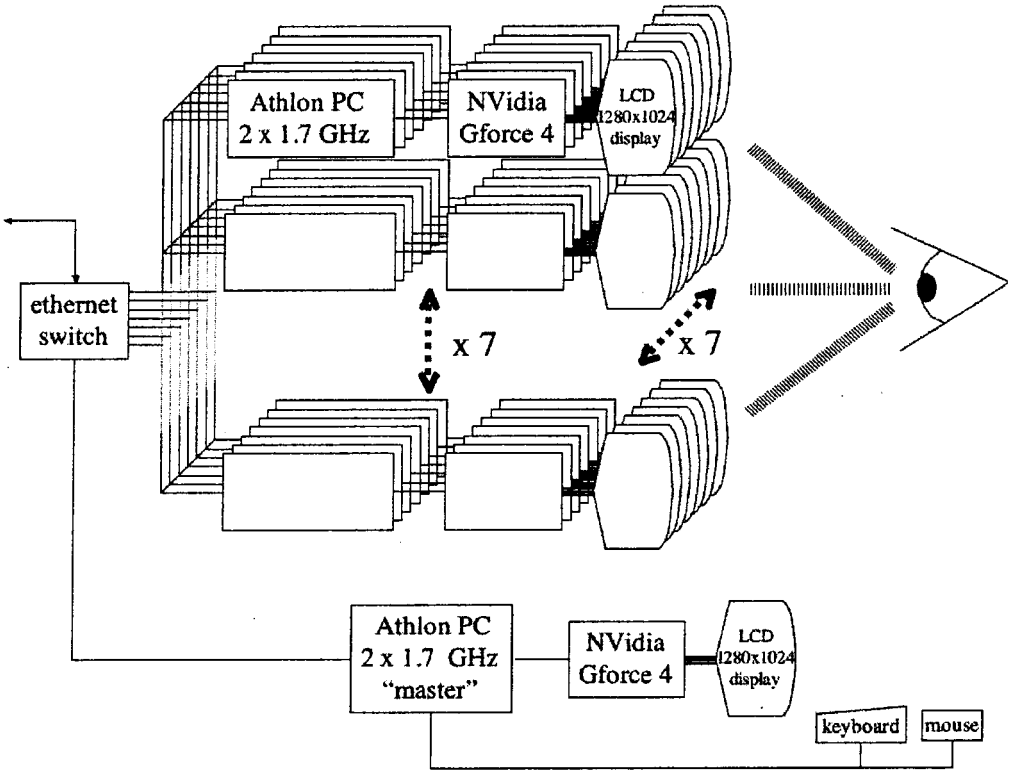

Figure 1: Architecture.

Figure 2: Examining an RLV parameter study.

face being rendered. Its frame rate drops to say, 3 fps while the other nodes gallop merrily along at 15 fps. The views will in this case diverge until the user releases the mouse, whereupon the views will again converge when all nodes in the gang have digested all the events in their identical event streams. If needed, a finer grained approach using some sort of distributed synchronization mechanism such as a barrier can remove this remaining issue, and provide frame for frame view coherence. This is needed for things such as synchronized animation.

4.3 Powerwall mode

Sooner or later, everyone eventually asks if we can show one image over all the screens (like a powerwall). The answer is yes, with some reservations. Unless you are using Chromium [10] to divide up of the graphics work, in general, node application software will need to be modified. We have used Chromium on our cluster but at the time, there were performance issues related to using only fast ethernet between nodes, as well as issues with display lists and applications using multiple windows.

For 2D scenes, such as rendering an image, there is an implied XY offset into the image based upon its location in the wall. Image rendering software would need to calculate the relevant subrectangle for its portion of the view, possibly affecting I/O, buffering, and display routines. Displaying 3D scenes across an array of screens can be achieved by dividing up the view frustum appropriately and having each node render the entire scene. While this seems wasteful, the efforts required for culling the scene via octrees, or other suitable schemes, usually does not become necessary until the rendered scenes become completely unwieldy. Modern graphic cards like the GeForce4 and its contemporaries are quite capable of rendering millions of polygons per second. Of course, one could come up with a giga-polygon isosurface to foil us here but in practice, this has not been an issue. View coherence is an issue however as discussed in section 4.2.

5 Navigating in Hyperspace

The complexity of MDMV datasets inspires us and cries out for novel tools to search around for new unseen relationships. Inevitably, there is the tension between complexity of task and simplicity of interface.

For example, navigation is one of the primary challenges in MDMV visualization. Successful interfaces for navigation are often intuitive, provide contextual information, and are flexible enough to get the user where they want to go. A good example of MDMV navigation is the HyperSlice [4] interface. The user is presented with an array of bivariate plots, each of which is centered on an n-dimensional point $\mathbf{C} = \langle \mathbf{X_1}, \mathbf{X_2}, ..., \mathbf{X_n} \rangle$ This point can be moved around in n-space by direct manipulation. By grabbing in the $\mathbf{X_1}, \mathbf{X_2}$ subplot the user changes those coordinates of C, leaving all the other dimensions $\mathbf{X_3}...\mathbf{X_n}$ constant. This interface gives immediate feedback to the user via direct manipulation, provides contextual information for the user through the meaningful layout of the subplots, and allows the user to navigate the center point C, anywhere in the n-dimensional space.

For our purposes, we have several datasets requiring navigation through a 6D space. One example, is the visualization of the electron pair density function. Given a molecule, we consider an nearby electron. Then, for each possible position of that electron in 3-space around the molecule, we consider all possible positions of a companion electron. With 3 degrees of freedom for each electron, this results in the 6D electron-pair density function which we wish to explore in order to find electron-rich or electron-poor areas.

For our purposes, it made sense to move a hyperplane around instead of a hyperpoint like the HyperSlice interface. On a given node, 3 dimensions of the data, say $\langle \mathbf{X}, \mathbf{Y}, \mathbf{Z} \rangle$, can be represented with volume rendering or some other suitable 3D scalar visualization technique. Then, with a navigational tool running on the master, we can interactively assign the remaining 3 dimensions $\langle \mathbf{U}, \mathbf{V}, \mathbf{W} \rangle$ by orienting a plane in 3D. On the plane are arranged an array of points, each of which represents a node in the cluster (and hence, a screen on the wall) and has a unique $\langle \mathbf{U}, \mathbf{V}, \mathbf{W} \rangle$ coordinate.

If the nodes are running a simulation, then the coordinates are typically continuous and are sent to the nodes to update their displays. Thus for example, each node would do a volume rendering of all $\langle \mathbf{X}, \mathbf{Y}, \mathbf{Z} \rangle$ data for a fixed $\langle \mathbf{U}, \mathbf{V}, \mathbf{W} \rangle$. For the situation when the $\langle \mathbf{U}, \mathbf{V}, \mathbf{W} \rangle$ coordinates are discrete the coordinates for a given node would be 'snapped' to the nearest valid coordinate. An example might be data files associated with a parameter study where a coordinate change might require the nodes to load a different file.

6 Interactive Parameterized Simulations

With the advent of workstation class PCs, significant computational power is availible to throw at a problem. When combined with an array of graphics displays, we approach an environment where we can exercise computational steering[9].

Our system allows us to run parameterized families of simulations in parallel. However, computational steering environments often require that one 'instrument' the simulation code in order to give the user feedback (monitoring) and allow interactive control (steering). For monitoring, the simulation code is often modified to allow display of the state of the simulation. Another modification would be to allow user access to the simulation's runtime parameters

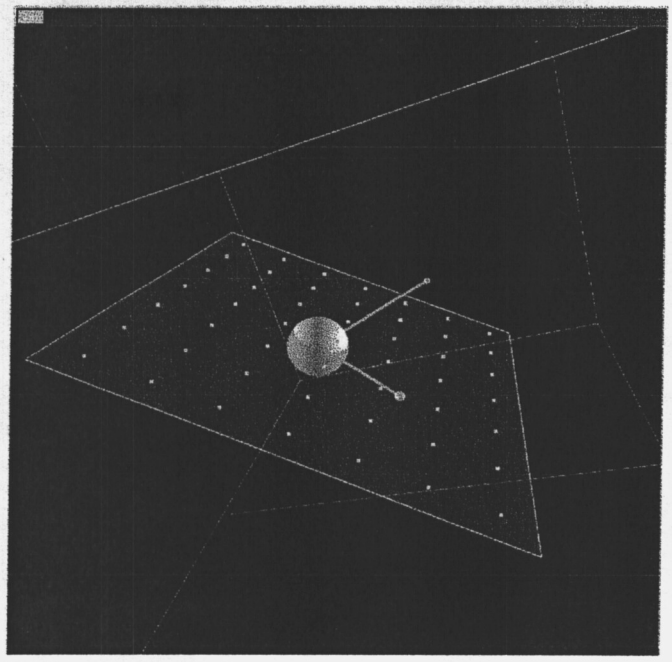

Figure 3: Six-D browser interface here shown with a schematic water molecule. This interface allows the user to coordinate the simulations seen in Figure 4.

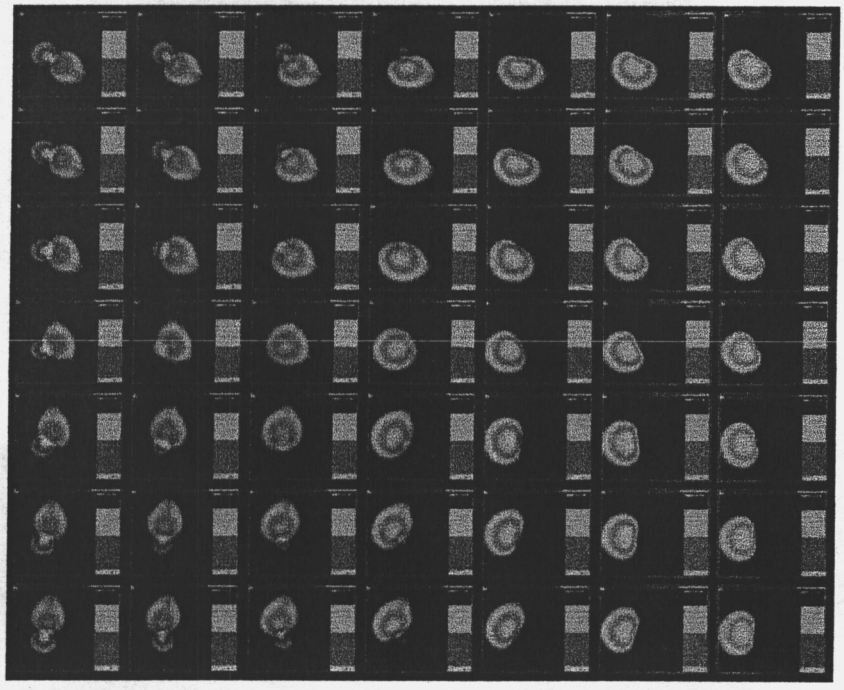

Figure 4: The Six-D browser allows interactive exploration of the electron pair density function around a water molecule.

for steering. Both of these modifications are delicate and obviously require in-depth analysis of any simulation code.

As an example, a molecular simulation code known as COSMOS (COmputer Simulations of MOlecular Systems) has been instrumented by NASA researchers Chris Henze and Brian Green. Every time COSMOS completes a timestep, it sends the updated atom positions to a viewer. Within the viewer, the user can interact with the simulation by selecting individual molecules and moving them around. Furthermore, using an interface on the master node, forces such as compression, expansion, rotation, and shear can be applied.

Within the hyperwall environment, dozens of these simulations can be run in parallel. Fig 5 displays thumbnail snapshots from a set of carbon nanotube simulations each of which has been subjected to an expansive force interactively supplied by the user. Again, layout in the matrix implies position in this parameter space of nanotubes. Along the diagonal, from top to bottom, the nanotubes grow larger. Greater distance above and below the diagonal corresponds to more twist (also known as the chirality number) in a clockwise or counterclockwise direction. We can see that, in general, the smaller tubes tend to break apart with the level of force applied by the user.

7 Conclusion

We have presented the hyperwall, a system capable of interactive exploration of MDMV data and simulations. Because we have a full blown beowulf cluster, each node of which is armed with its own graphics card, we can run compute and graphics intensive applications, bringing a powerful array of tools to bear on any given problem. This allows us to compute whole arrays of visualzations or simulations in parallel, displayed at high resolution, in a highly interactive fashion. The sheer visual nature of the display system encourages people to scan the displays looking for trends, relationships, and anomilies. We find that scientists want to walk right up to the wall of screens, look closer, and point out observational curiosities to co-investigators making it inherently collaborative in nature.

Acknowledgements

This work was sponsored by NASA contract TOA61812D. Thanks to David Ellsworth, for all the encouragement and helpful insights.

References

[1] J. W. Tukey. "Exploratory Data Analysis," Addison-Wesley, 1977

[2] R. A. Becker and W. S. Cleveland. "Brushing Scatterplots," Technometrics, 29(2), 127–142, 1987

[3] D. F. Swayne, D. Cook, A. Buja. "XGobi: Interactive Dynamic Graphics in the X Window System with a Link to S," in ASA Proceedings of the Section on Statistical Graphics, p. 1–8, 1991

[4] J. van Wijk, R. van Liere. "Hyperslice - visualization of scalar functions of many variables," In Proceedings of IEEE Visualization, 1993

[5] M. Ward, "XmdvTool: integrating multiple methods for visualizing multivariate data," In Proceedings of IEEE Visualization 1994, pp. 326–333

[6] M. Levoy. "Spreadsheet for images," In Computer Graphics (SIGGRAPH '94 Proceedings), volume 28, pages 139–146. SIGGRAPH, ACM Press, 1994

[7] A. Varshney and A. Kaufman. "FINESSE: A financial information spreadsheet," In IEEE Information Visualization Symposium, pages 70–71, 125, 1996

[8] Ed H. Chi, J. Riedl, P. Barry, J. Konstan. "Principles for Information Visualization Spreadsheets," In IEEE Computer Graphics and Applications (Special Issue on Visualization) July/August, 1998. IEEE CS, pp. 30–38.

[9] J. Mulder, J. van Wijk, and R. van Liere. "A Survey of Computational Steering Environments," in Future Generation Computer Systems, 13(6), 1998

[10] G. Humphreys, M. Houston, R. Ng, R. Frank, S. Ahern, P. Kirchner, J. T. Klosowski. "Chromium: A Stream-Processing Framework for Interactive Rendering on Clusters," presented at SIGGRAPH, San Antonio, Texas, 2002

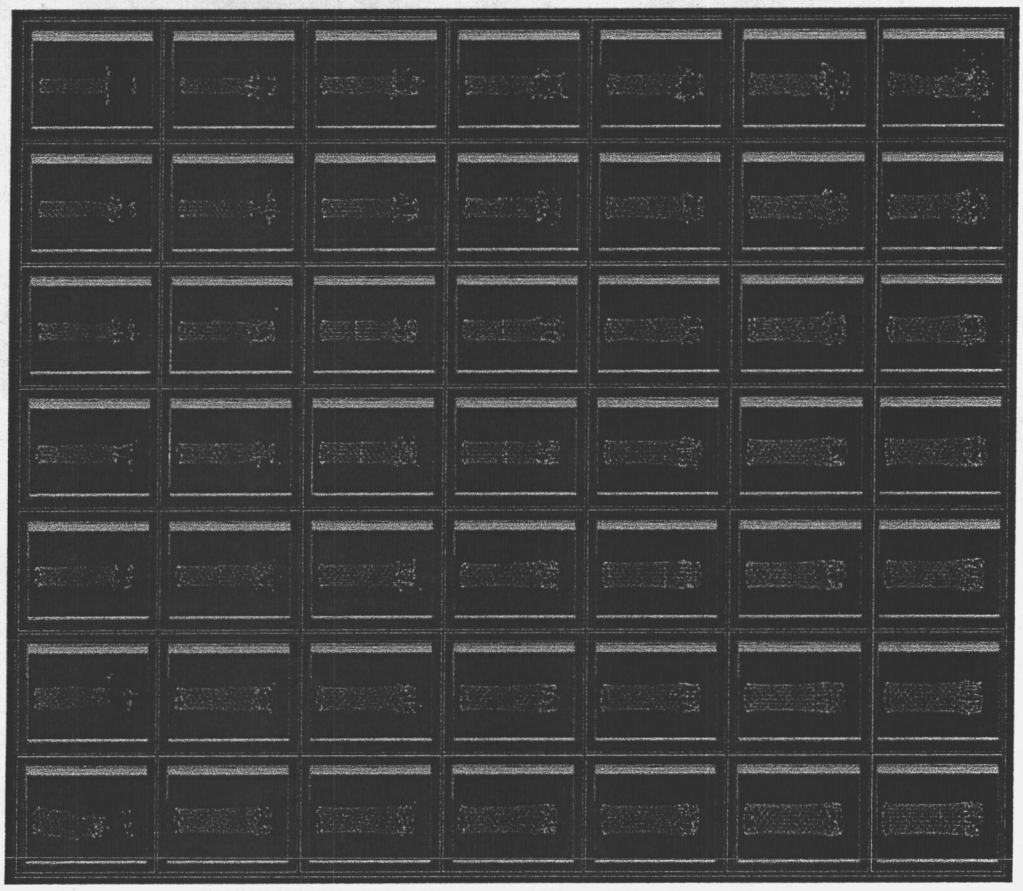

Figure 5: Realtime interaction with a family of nanotube simulations.